Розглянуто особливості формування числової моделі джерела даних. Досліджено вплив на коефіцієнт ущільнення довжини блоків даних, на які розбивається ущільнюваний файл. Запропоновано і досліджено два методи адаптивного ущільнення даних на основі лінійної форми Фібоначчі, які передбачають використання набору числових моделей джерела даних

Ключові слова: адаптивне ущільнення, числова модель, джерело даних, лінійна форма Фібоначчі,коефіцієнт ущільнення

Рассмотрены особенности формирования числовой модели источника данных. Исследовано влияние на коэффициент сжатия длины блоков данных, на которые разбивается сжимаемый файл. Предложено и исследовано два метода адаптивного сжатия данных на основе линейной формы Фибоначчи, предусматривающие использование набора числовых моделей источника данных

Ключевые слова: адаптивное сжатие, числовая модель, источник данных, линейная форма Фибоначчи, коэффициент сжатия

УДК 519.6: 621.391
DOI: 10.15587/1729-4061.2015.37026

РОЗРОБКА ТА ДОСЛІДЖЕННЯ МЕТОДІВ АДАПТИВНОГО УЩІЛЬНЕННЯ ДАНИХ НА ОСНОВІ ЛІНІЙНОЇ ФОРМИ ФІБОНАЧЧІ

В. А. Лужецький
Доктор технічних наук,
професор, завідувач кафедри
Кафедра захисту інформації*
E-mail: lva_zi@mail.ru
Л. А. Савицька
Асистент
Кафедра обчислювальної техніки*
E-mail: Lyudik0304@gmail.com
*Вінницький національний технічний університет
Хмельницьке шосе, 95, м. Вінниця, Україна, 21021

1. Вступ

Інформаційний вибух останніх років привів до значного зростання обсягів інформації, що передається та оброблюється в комп'ютерних системах і мережах. Це вимагає збільшення обсягів пам'яті та канальних ресурсів. У свою чергу, структурне або апаратне збільшення призводить до зменшення продуктивності системи. Одним із підходів щодо подолання цієї проблеми є використання засобів ущільнення інформації (компресорів і архіваторів). Ущільнення інформації скорочує обсяг пам'яті, що необхідна для її зберігання, і кількість часу, який потрібен для її передавання каналом з фіксованою пропускною здатністю. За останнє десятиліття продуктивність процесорів зросла експоненційно порівняно зі швидкістю доступу до пристроїв пам'яті, що є ґрунтовною причиною застосовувати ущільнення інформації для збільшення загальної продуктивності системи.

Існує кілька підходів до ущільнення інформації, що породжують цілу низку методів ущільнення [1–6], для реалізації яких, у свою чергу, використовується величезна кількість алгоритмів. Одні з них мають науковий характер, а інші знайшли практичну реалізацію у вигляді компресорів і архіваторів.

Теоретичні дослідження і практика застосування архіваторів показали, що не існує універсального методу ущільнення, що забезпечував би однаковий степінь ущільнення для різних типів даних. Тому продовжуються пошуки нових підходів, що були б ефективними для певних типів сигналів і даних з точок зору степеня ущільнення, програмної й апаратної реалізації.

2. Аналіз літературних даних та постановка проблеми

Одним з найважливіших положень теорії ущільнення інформації є висловлена в [7] ідея поділу процесу ущільнення на дві процедури: моделювання і кодування.

Моделювання визначає характеристики джерела даних, що ущільнюються. Залежно від того, якою моделлю джерела сигналу описується інформація, її поділяють на бінарну інформацію (виконувані файли, програмні бібліотеки і т.д.) і текстову. Окремим випадком текстової або бінарної інформації є числові дані (залежно від того, у якому вигляді вони представлені – ASCII коді або двійковому).

Метою кодування є перетворення потоку символів у потік біт мінімальної довжини. У роботі [8] наведено таку класифікацію методів кодування:

– статистичний (у цьому методі припускається відповідність вхідного потоку певній моделі сигналу і здійснюється ущільнення на основі зібраної про текст статистичної інформації);

– макро- або текстової підстановки (метод базується на відшуканні однакових рядків і заміні їх на більш короткі коди);

– інкрементний (ущільнення здійснюється шляхом кодування відмінностей у послідовних записах).

Для реалізації ефективного кодування необхідно певним чином вибрати алфавіт і відповідний розподіл символів у потоці. Це забезпечується побудовою моделі вхідного потоку, яка описує деякий спосіб визначення можливого розподілу ймовірностей появи кожного чергового символу в потоці. Якщо розподіл ймовірностей частот появи символів з алфавіту вхідного потоку відомий, то можна побудувати модель оптимального кодування. Однак через існування величезної кількості різних форматів файлів задача значно ускладнюється. Статичні моделі, в яких розподіл приймається незмінним, у більшості випадків, не забезпечують максимального степеня ущільнення. Набагато більший інтерес мають, так звані, адаптивні моделі, які враховують потоковий контекст потоку.

Окрему групу методів ущільнення даних без втрат складають методи, що базуються на кодуванні цілих чисел з використанням нерівномірних кодів Еліаса [9], Райса [10], Левенштейна [11], Голомба [12], Івен-Роде [13], Фібоначчі [14, 15] та інших.

Останнім часом інтенсивно досліджуються можливості практичного використання кодів Фібоначчі для ущільнення даних. Є пропозиції щодо зміни стандартного формату JPEG шляхом заміни статистичного кодування на кодування Фібоначчі [16]. В роботі [17] показано, що застосування кодів Фібоначчі для ущільнення геофізичної інформації забезпечує коефіцієнт ущільнення від 10 % до 30 % більше порівняно з коефіцієнтами, що забезпечують коди Хаффмана, Еліаса, Райса і Голомба.

Відзначимо, що перевага вищеописаних кодів полягає у тому, що ущільнення даними методами не вимагає великої кількості пам'яті, а також однаково швидко виконується як ущільнення, так і відновлення інформації.

І хоча на практиці широко використовуються архіватори, створені на основі методів й алгоритмів ущільнення даних, що враховують статистику символів у повідомленні [1–3] або базуються на побудові словника [4–6], однак продовжуються пошуки нових підходів до ущільнення даних.

У роботах [18, 19] запропоновано оригінальний спосіб представлення цілих чисел у вигляді, так званої, лінійної форми Фібоначчі, яка забезпечує скорочення розрядності представлення великих чисел (1024 двійкові розряди і більше).

Лінійна форма Фібоначчі рангу t – це представлення цілого числа із застосуванням чисел Фібоначчі, що має вигляд [18]:

$$N = xF_{t-1} + yF_t, \qquad (1)$$

де x і y – цілі числа, $y \neq 0$; t – натуральне число.

Така форма представлення є основою нового підходу щодо ущільнення даних, який ґрунтується на оптимізуючих властивостях чисел Фібоначчі [26]. При цьому блок цифрових даних розглядається як ціле додатне число N.

Існує множина представлень виду (1), але для задачі ущільнення даних пропонується використовувати представлення максимального рангу, яке має найменші значення цілих додатних чисел x і y. Тільки в цьому випадку з'являється можливість максимально використовувати оптимізаційні властивості чисел Фібоначчі.

Задача ущільнення полягає у знаходженні для заданого числа N мінімального базису (a_0, b_0) такого, що $N = \Phi_j(a_0, b_0)$. Числа $\Phi_i(a_0, b_0)$ – це узагальнені числа Фібоначчі, що обчислюються за формулою [18]:

$$\Phi_i = \Phi_{i-1} + \Phi_{i-2}, \ i = 2 \div j \qquad (2)$$

для $\Phi_0 = a_0$; $\Phi_1 = b_0$.

У роботі [20] для конкретних типів файлів показано, що існує принципова можливість ущільнення даних з використанням лінійної форми Фібоначчі. Однак відсутні дослідження цього методу стосовно ефективності ущільнення певних типів даних. Крім того, не досліджено вплив розрядності ущільнюваного блоку даних на коефіцієнт ущільнення.

Теоретичні дослідження і практика застосування відомих архіваторів показали, що не існує універсального методу ущільнення, який забезпечував би однаковий степінь ущільнення для різних типів даних. Тому дослідження науковців спрямовані на створення ефективних методів ущільнення певних типів даних. Однак дані лише одного типу, з точки зору ущільнення, мають різні властивості й характеристики. Враховуючи це, останнім часом прагнуть до створення адаптивних алгоритмів ущільнення даних.

3. Мета та задачі дослідження

Метою дослідження є підвищення степеня ущільнення даних на основі лінійної форми Фібоначчі шляхом розробки адаптивних методів ущільнення.

Для досягнення поставленої мети розв'язувалися такі задачі:

– дослідження впливу на коефіцієнт ущільнення довжини блоків даних, на які розбивається ущільнюваний файл;

– розробка і дослідження методів адаптивного ущільнення даних на основі лінійної форми Фібоначчі.

4. Методика та результати дослідження впливу довжини блоків даних на коефіцієнт ущільнення.

Вихідні дані, що підлягають ущільненню, будемо розглядати як послідовність символів 0 і 1. Відповідно до числової моделі джерела даних, послідовність символів розбивається на блоки, що містять деяку кількість символів, і кожному символу в блоці відповідає своя певна вага. Якщо вага k-го символу блоку дорівнює (k=0,1..., n–1), то блок є двійковим кодом деякого числа.

Для практичної реалізації більш зручним є представлення числа сукупністю байтів. При цьому кожен байт – це відповідний ASCII код, числовий еквівалент якого визначається за формулою:

$$s = \sum_{i=0}^{7} a_i 2^i, \qquad (3)$$

де a_i – цифра i-го розряду ASCII коду.

Виходячи з цього, блок даних довжиною *l* байтів представляється як послідовність *l* чисел:

$$s_0, s_1, s_2, ..., s_{l-1}, \quad (4)$$

де s_j – числовий еквівалент ASCII коду *j*-го байту (j=0,1,2,...,l-1).

Оскільки числові еквіваленти ASCII кодів належать діапазону чисел від 0 до 255, то число, що відповідає блоку даних, може бути обчислено за формулами:

$$N_м = \sum_{j=0}^{l-1} s_j 256^j, \quad (5)$$

$$N_c = \sum_{j=0}^{l-1} s_{(l-j-1)} 256^{(l-j-1)}. \quad (6)$$

Відмінність цих формул полягає тільки в порядку використання елементів послідовності (4) відносно їх номерів: від молодшого до старшого та навпаки. Тут число 256 є основою системи числення і для кожного блоку даних вона незмінна (фіксована). Виходячи з цього, позначимо моделі джерела даних таким чином: $M_{фм}$ і $M_{фс}$, відповідно.

Зміною кількості байтів у блоці забезпечується отримання послідовностей чисел з різними законами їхнього формування. Тобто для різних *l* будемо мати різні моделі джерела даних. Можливість зміни моделі джерела даних дозволяє вибирати таку модель, яка забезпечує найбільший коефіцієнт ущільнення для заданого правила кодування.

Найчастіше коефіцієнт ущільнення визначається за формулою:

$$S_{ущ.} = \frac{L_0}{L_{ущ.}}, \quad (7)$$

де L_0 – довжина послідовності даних до ущільнення в байтах; $L_{ущ.}$ – довжина послідовності даних після ущільнення в байтах.

Для блоків довжини, яких змінюються від l_{min} до l_{max} з певним кроком, виконується ущільнення і вибирається та довжина, при якій забезпечується найбільший коефіцієнт ущільнення.

Правило вибору з множини кодованих послідовностей $\mathbf{P}_C$ єдиної послідовності $\mathbf{P}^*$, що відповідає найбільшому коефіцієнту ущільнення, описується функцією оптимізації на рівні послідовностей, яка має вигляд:

$$f_{посл} = \min\{L_{ущ}^{(m)}\}, \quad (8)$$

де $L_{ущ}^{(m)}$ – довжина послідовності ущільнених даних для *m*-ої моделі джерела даних.

Дослідження здійснювалося з використанням програмного засобу, що реалізує метод ущільнення, суть якого полягає в такому. Ущільнювані дані **P** розбиваються на блоки довжини l байтів. У загальному випадку, останній (k-й) з послідовності блоків може мати довжину $l_k < l$. Для кожного блоку визначається числовий еквівалент N за формулою (5), в разі моделі джерела даних $M_{фм}$, або за формулою (6), коли використовується модель джерела даних $M_{фс}$. Якщо N=0, то формується тільки ознака p=0, а якщо N>0, то формується ознака p=1 і здійснюється перетворення числа N в лінійну форму Фібоначчі:

$$N = a_0 F(j) + b_0 F(j+1), \quad (9)$$

де $F(j)$ – j-е число Фібоначчі.

Ущільнені дані $\mathbf{P}^*$ мають таку структуру:

$$S=\{l||l_k||\pi||Бл^*1||Бл^*2||...||Бл^*k\}, \quad (10)$$

де l – значення довжини блоку (2 байти); l_k – значення довжини останнього блоку (2 байти); π – послідовність ознак p у байтовому представленні; Бл*i – структура складових лінійної форми Фібоначчі числового еквівалента *i*-го блоку ($i = 1,2,...,k$).

Структура Бл*i така:

$$\{j||l_{a_0}||l_{b_0}||a_0||b_0\}, \quad (11)$$

де j – значення числа j (2 байти); l_{a_0} – довжина коду числа a_0 в байтах (2 байти); l_{b_0} – довжина коду числа b_0 в байтах (2 байти); a_0 – код числа a_0; b_0 – код числа b_0.

Довжина блоку l змінювалася в межах від l_{min} = =100 байт до l_{max} =1000 байт з кроком 100 байт.

Оскільки числові еквіваленти блоків даних мають великі значення (від 2^{800} до 2^{8000}), то програмний засіб було створено мовою програмування Python.

Досліджувалося ущільнення файлів, тип та обсяг яких наведено в табл. 1.

Аналіз отриманих результатів показав, що ущільнюються файли типів *.doc, *.dat, *.bmp, *.lib, *.mp3, *.mdb, *.dll. Обсяги ущільнених файлів наведено в табл. 2, 3. Тут жирним шрифтом виділено найменші обсяги.

Аналіз отриманих результатів показує, що вплив довжини блоків даних l на коефіцієнт ущільнення є різним для різних типів файлів. Найбільший коефіцієнт ущільнення для файлів d_20.doc і db_50.mdb досягається в разі довжини блоку 100 байт, а зі збільшенням довжини коефіцієнт зменшується. Для файлу bootstat.dat вплив довжини блоків даних має «хвильовий» характер (зі збільшенням довжини коефіцієнт кілька разів збільшується і зменшується) для моделі джерела даних $M_{фм}$. Найбільший коефіцієнт ущільнення досягається для довжини блоку 700 байт. У разі моделі джерела даних $M_{фс}$ найбільший коефіцієнт ущільнення досягається для довжини блоку 100 байт, а зі збільшенням довжини коефіцієнт зменшується.

Таблиця 1

Тип та обсяг досліджуваних файлів

Назва файлу	d_20. doc	bootstat.dat	b_100.bmp	m_40.mp3	db_50.mdb	idasci32.dll	bfcd40d.lib
L_0 (байт)	20480	67584	126920	38912	67584	95744	38912
Назва файлу	avtrans port. xml	g_20. gif	j_100. jpg	resfmt. txt	date. cpp	classic. kbd	au_160. au
L_0 (байт)	19842	19910	107808	46341	9544	52633	171100

Таблиця 2

Обсяги файлів, ущільнених з використанням моделі джерела даних $M_{фм}$

l (байт)	d_20. doc	bootstat. dat	b_100. bmp	m_40. mp3	db_50. mdb	idasci32.dll	bfcd40d. lib
100	**8607**	294	**116980**	39925	**24370**	98121	38531
200	8899	248	117401	38916	26333	95871	**38156**
300	9434	438	118529	38426	27266	95576	38279
400	9844	227	120443	38544	30146	94949	38208
500	10119	223	121384	38121	31450	**94555**	39051
600	10352	420	122948	38346	31237	95140	38164
700	11865	822	123471	**38087**	33744	94635	38518
800	10543	615	124071	38655	36815	94970	38419
900	12321	415	122930	38525	37128	95143	38878
1000	12818	**214**	124063	38393	38021	95332	39047

Таблиця 3

Обсяги файлів, ущільнених з використанням моделі джерела даних $M_{фс}$

l (байт)	d_20. doc	bootstat. dat	b_100. bmp	m_40. mp3	db_50. mdb	idasci32.dll	bfcd40d. lib
100	**8671**	**358**	116272	39986	**24253**	98445	38443
200	9730	411	**115672**	38944	25938	95947	38424
300	9125	703	117667	38456	28629	95783	**37905**
400	10473	790	118882	38375	28757	95203	38669
500	11531	986	120490	**38354**	31525	94977	38680
600	10794	983	120898	38376	33262	95497	38458
700	11458	1487	121981	38420	33473	95421	38251
800	12434	1179	122654	38485	34451	94725	38736
900	12936	1578	124548	38553	36933	95498	38687
1000	13253	1977	123782	38622	39074	**94634**	39131

Коефіцієнти ущільнення для файлів b_100.bmp, m_40.mp3, idasci32.dll і bfcd40d.lib є незначними і досягаються при певних довжинах блоків.

Обсяг перетворених файлів типів *.txt, *.xml, *.kbd, *.fdb, *.sdb, *.cpp, *.gif, *.jpg і *.au перевищує обсяг вихідних файлів. Однак спостерігається тенденція наближення обсягу перетворених файлів до обсягу вихідних фалів при збільшенні довжини блоків даних.

Порівняння результатів ущільнення файлів з використанням двох різних моделей джерела даних показує, що модель $M_{фм}$ є більш адаптованою для файлів d_20.doc, bootstat.dat, m_40.mp3 та idasci32.dll, а модель $M_{фс}$ - для файлів b_100.bmp, db_50.mdb і bfcd40d.lib, оскільки саме вони забезпечують найменший обсяг ущільнених даних. Для файлів, що не ущільнюються, залежність обсягу перетворених даних від використовуваної моделі джерела даних є незначною.

5. Методи адаптивного ущільнення даних

Пропонується метод ущільнення, суть якого полягає в такому. Ущільнювані дані **P** розбиваються на блоки довжини l байтів. Для кожного блоку визначаються два числові еквіваленти $N_м$ і $N_с$: один за формулою (5), а другий за формулою (6). Якщо $N_м=N_с=0$, то формується тільки ознака $p=0$, а інакше, формується ознака $p=1$ і здійснюється перетворення чисел $N_м$ і $N_с$ в лінійну форму Фібоначчі ЛФФ($N_м$) і ЛФФ($N_с$). Якщо в байтовому представленні $ЛФФ(N_м) \le ЛФФ(N_с)$, то формується ознака $c=0$ і результатом перетворення блоку є ЛФФ($N_м$), а інакше, формується ознака $c=1$ і результатом перетворення блоку є ЛФФ($N_с$). Тобто оптимізація здійснюється на рівні блоків.

При цьому функція оптимізації має вигляд:

$$f_{бл}^{оп} = \min\{l_{бл}^{(фм)}, l_{бл}^{(фс)}\}, \quad (12)$$

де $l_{бл}^{(фм)}$ – довжина перетвореного блоку для моделі джерела даних $M_{фм}$; $l_{бл}^{(фс)}$ – довжина перетвореного блоку для моделі джерела даних $M_{фс}$.

Ущільнені дані $\mathbf{P}^*$ мають таку структуру:

$$S=\{l||l_k||\pi||C||Бл^*1||Бл^*2||...||Бл^*k\}, \quad (13)$$

де C – послідовність ознак *c* у байтовому представленні.

Для зручності позначимо комбінацію двох моделей джерела даних як модель $M_{фмс}$.

Результати дослідження цього методу ущільнення наведено в табл. 4.

Порівняння результатів, наведених у табл. 2–4, показує, що модель джерела даних $M_{фмс}$ забезпечує менший обсяг для всіх ущільнених файлів. Однак ця модель, як і дві попередні, є неефективною для файлів типу *.txt, *.xml, *.kbd, *.fdb, *.sdb, *.cpp, *.gif , *.jpg і *.au.

Незначні коефіцієнти ущільнення і відсутність ущільнення для певних типів файлів пояснюється тим, що використовуване моделювання даних не забезпечує формування чисел, які компактно представляються лінійною формою Фібоначчі. З цього випливає необхідність підвищення адаптованості перетворень до змісту файлів шляхом збільшення кількості моделей джерела даних.

Таблиця 4

Обсяги файлів, ущільнених з використанням моделі джерела даних $M_{фмс}$

l (байт)	d_20. doc	bootstat. dat	b_100. bmp	m_40. mp3	db_50. mdb	idasci32.dll	bfcd40d. lib
100	**7888**	250	114112	39916	**21629**	97547	37591
200	8229	236	**113068**	38794	23015	95265	37128
300	8043	194	114117	38402	23648	94990	**36819**
400	8207	228	115363	38222	24179	94362	37438
500	8689	224	118075	38096	25817	**94003**	38352
600	8831	408	119523	38020	26639	94542	37564
700	9377	**178**	120664	37960	26251	94196	37660
800	9950	404	121277	37930	27107	94082	37946
900	10470	416	120927	37897	27270	94256	38406
1000	9503	215	122521	**37865**	28306	93777	39028

Для розв'язання цієї задачі пропонується визначати числовий еквівалент кожного блоку даних виходячи з максимального значення числових еквівалентів ASCII кодів байтів, що складають цей блок.

Аналіз змісту блоків ущільнюваних даних стосовно числових еквівалентів ASCII кодів показує, що не в кожному блоці є байт з числовим еквівалентом 255. Тому є можливість визначати числовий еквівалент блоку через представлення в системі числення з основою не тільки 256.

Покажемо це на прикладі. Нехай блок даних складається з байтів з такими числовими еквівалентами: 117; 45; 67; 136. Тут найбільшим є число 136. Отже, можна визначити числовий еквівалент блоку, використовуючи представлення в системах числення з основами від 137 до 256. Тобто для такого блоку можна отримати 120 різних числових значень.

Виходячи з цього, пропонується такий метод адаптивного ущільнення даних.

Ущільнювані дані **P** розбиваються на блоки довжини l байтів. Для кожного блоку визначається максимальне значення B_{max} серед значень числових еквівалентів байтів. Потім обчислюється два числові еквіваленти $N_м$ і $N_с$: один за формулою:

$$N_м = \sum_{j=0}^{l-1} s_j B^j , \qquad (14)$$

а другий за формулою:

$$N_с = \sum_{j=0}^{l-1} s_{(l-j-1)} B^{(l-j-1)} , \qquad (15)$$

де $B = B_{max} + 1$.

Тут число B є основою системи числення і для кожного блоку даних воно змінюється від $(B_{max}+1)$ до 256. Виходячи з цього, позначимо моделі джерела даних таким чином: $M_{зм}$ і $M_{зс}$, відповідно.

Якщо $N_м$=$N_с$=0, то формується тільки ознака p=0, а інакше, формується ознака p=1 і здійснюється перетворення чисел $N_м$ і $N_с$ в лінійну форму Фібоначчі ЛФФ($N_м$) і ЛФФ($N_с$). Якщо в байтовому представленні $ЛФФ(N_м) \le ЛФФ(N_с)$, то формується ознака c=0 і результатом перетворення блоку є ЛФФ($N_м$), а інакше, формується ознака c=1 і результатом перетворення блоку є ЛФФ($N_с$).

Обчислення числових еквівалентів блоку $N_м$ та $N_с$ і перетворення їх у лінійну форму Фібоначчі виконується для всіх значень B, що не перевищують 256. Серед лінійних форм Фібоначчі вибирається така, що утворює перетворений блок найменшої довжини. При цьому фіксується значення B, яке є оптимальним (B_o). Тобто оптимізація здійснюється на рівні блоків. При цьому функція оптимізації має вигляд:

$$f_{бл}^{оп} = \min\left\{(\min\{l_{бл}^{(м)}, l_{бл}^{(с)}\})^{(i)}\right\}, \qquad (16)$$

де $l_{бл}^{(зм)}$ – довжина перетвореного блоку для i-ої моделі джерела даних $M_{зм}$; $l_{бл}^{(зс)}$ – довжина перетвореного блоку для i-ої моделі джерела даних $M_{зс}$.

Для зручності позначимо комбінацію моделей джерела даних як модель $M_{змс}$.

Ущільнені дані $\mathbf{P}^*$ мають структуру вигляду (13). Відмінність полягає тільки в структурі *Бл*i*, яку доповнено значенням B_o. Структура *Бл*i* має такий вигляд:

$$\left\{B_o \| j \| l_{a_0} \| l_{b_0} \| a_0 \| b_0\right\}. \qquad (17)$$

Результати ущільнення за даним методом наведено в табл. 5–7.

У табл. 5 можна побачити, що зменшення кроку зміни довжини блоку забезпечує отримання менших обсягів ущільнених даних.

Таблиця 5

Результати ущільнення файлів з використанням моделі джерела даних $M_{змс}$

d_20.doc		db_50.mdb		bootstat. dat	
l (байт)	$L_{ущ}$ (байт)	l (байт)	$L_{ущ}$ (байт)	l (байт)	$L_{ущ}$ (байт)
40	7746	40	20801	640	291
60	7672	60	**20478**	660	230
80	**7619**	80	20588	680	172
100	7656	100	20753	700	169
120	7743	120	20989	720	**167**
140	7774	140	20981	740	216

У табл. 7 наведено найменші обсяги перетворених файлів для довжини блоку 1000 байт.

Аналіз табл. 7 показує, що файли avtransport.xml, resfmt.txt, date.cpp і classic.kbd, які не ущільнювалися з використанням інших моделей джерела даних, ущільнені за даним методом. Проте для файлів g_20.gif, j_100.jpg та au_160.au модель джерела даних $M_{змс}$ також є неефективною.

Таблиця 6

Обсяги файлів, ущільнених з використанням моделі джерела даних $M_{змс}$

Назва файлу	b_100. bmp	m_40. mp3	idasci32.dll	bfcd40d. lib
L_0 (байт)	126920	38912	95744	38912
$L_{ущ}$ (байт)	**107667**	**37871**	**93745**	**36628**
l (байт)	200	1000	500	300

Таблиця 7

Обсяги файлів, перетворених з використанням моделі джерела даних $M_{змс}$

Назва файлу	avtrans port. xml	g_20. gif	j_100. jpg	resfmt. txt	date. cpp	classic. kbd	au_160.au
L_0 (байт)	19842	19910	107808	46341	9544	52633	171100
$L_{ущ}$ (байт)	**17313**	20059	108592	**40633**	**8400**	**46252**	172033

6. Порівняльний аналіз методів ущільнення даних

У табл. 8 наведено значення коефіцієнтів ущільнення за методами, що передбачають використання неадаптивних ($M_{фм}$ і $M_{фс}$) та адаптивних ($M_{фмс}$ і $M_{змс}$) моделей джерела даних.

Таблиця 8

Значення коефіцієнтів ущільнення

Назва файлу	Модель джерела даних			
	$M_{фм}$	$M_{фс}$	$M_{фмс}$	$M_{змс}$
d_20.doc	2,38	2,36	2,6	2,69
bootstat.dat	315,8	188,8	379,7	404,7
b_100.bmp	1,08	1,09	1,12	1,18
m_40.mp3	1,02	1,01	1,03	1,03
db_50.mdb	2,77	2,79	3,12	3,30
idasci32.dll	1,01	1,01	1,02	1,02
bfcd40d.lib	1,02	1,03	1,06	1,06
avtransport.xml	0,994	0,994	0,994	1,15
resfmt.txt	0,994	0,994	0,994	1,14
date.cpp	0,993	0,993	0,993	1,14
classic.kbd	0,994	0,994	0,994	1,14

Модель $M_{фм}$ має перевагу за модель $M_{фс}$ для файлів d_20.doc, bootstat.dat і m_40.mp3 та поступається їй для файлів b_100.bmp, db_50.mdb і bfcd40d.lib. Для решти файлів ці моделі рівноцінні. Моделі $M_{фм}$, $M_{фс}$ і $M_{фмс}$ забезпечують отримання коефіцієнтів ущільнення більших за одиницю для 7 файлів з 11. Модель $M_{змс}$ забезпечує найбільші коефіцієнти ущільнення для всіх файлів, але найефективнішою вона є для файлів bootstat.dat, db_50.mdb і d_20.doc. Варто відзначити, що архіватор RAR ущільнює файл bootstat.dat з коефіцієнтом 282,8 проти 404,7 в даному випадку.

7. Висновки

1. Методи ущільнення даних на основі лінійної форми Фібоначчі передбачають використання числової моделі джерела даних, відповідно до якої ущільнювані дані розбиваються на блоки певної довжини. Зміною кількості байтів у блоці забезпечується отримання послідовностей чисел з різними законами їхнього формування. Можливість зміни моделі джерела даних дозволяє вибирати таку модель, яка забезпечує найбільший коефіцієнт ущільнення для заданого правила кодування. Аналіз результатів досліджень показав, що вплив довжини блоків даних на коефіцієнт ущільнення є різним для різних типів файлів. Незначні коефіцієнти ущільнення і відсутність ущільнення для певних типів файлів пояснюється тим, що використовуване моделювання даних не забезпечує формування чисел, які компактно представляються лінійною формою Фібоначчі. З цього випливає необхідність підвищення адаптованості перетворень до змісту файлів шляхом збільшення кількості моделей джерела даних.

2. Аналіз змісту блоків ущільнюваних даних стосовно числових еквівалентів ASCII кодів показав, що не в кожному блоці є байт з числовим еквівалентом 255. Тому є можливість обчислювати числовий еквівалент блоку через представлення в системі числення з основою не тільки 256, що збільшує кількість числових еквівалентів. Для цього потрібно визначати максимальне значення числових еквівалентів ASCII кодів байтів, що складають блок. Адаптація забезпечує підвищення коефіцієнта ущільнення порівняно з неадаптивним методом ущільнення на основі лінійної форми Фібоначчі, який передбачає тільки одну модель джерела даних для кожного блоку вхідних даних. Модель $M_{змс}$ забезпечує найбільші коефіцієнти ущільнення для всіх досліджених файлів, але найефективнішою вона є для файлів bootstat.dat, db_50.mdb і d_20.doc.

Література

1. Shannon, C. E. A Mathematical Theory of Communication [Text] / C. E. Shannon // Bell System Technical Journal. – 1948. – Vol. 27, Issue 3. – P. 379–423. doi: 10.1002/j.1538-7305.1948.tb01338.x
2. Huffman, D. A. A Method for the Construction of Minimum-Redundancy Codes [Text] / D. A. Huffman // Proceedings of the Institute of Electrical and Radio Engineers. – 1952. – Vol. 40, Issue 9. – P. 1098–1101. doi: 10.1109/jrproc.1952.273898

3. Witten, I. Arithmetic Coding for Data Compression [Text] / I. Witten, R. Neal, J. Cleary // Communications of the ACM. – 1987. – Vol. 30, Issue 6. – P. 520–540. doi: 10.1145/214762.214771
4. Ziv, J. A universal algorithm for sequential data compression [Text] / J. Ziv, A. Lempel // IEEE Transactions on Information Theory. –1977. – V. 23, №3. – P. 337–343.
5. Ziv, J. Compression of individual sequences via variable-rate coding [Text] / J. Ziv, A. Lempel // IEEE Transactions on Information Theory. – 1978. – Vol. 24, Issue 5. – P. 530–535. doi: 10.1109/tit.1978.1055934
6. Welch, T. A. A Technique for High Performance Data Compression [Text] / T. A. Welch // Computer. – 1984. – Vol. 17, Issue 6. – P. 176–189. doi: 10.1109/mc.1984.1659158
7. Rissanen, J. J. Universal modeling and coding [Text] / J. J.Rissanen, G. G. Langdon // IEEE Transactions on Information Theory. – 1981. – Vol. 27, Issue 1. – P. 12–23. doi: 10.1109/tit.1981.1056282
8. Storer, J. A. Data compression via textual substitution [Text] / J. A. Storer, T. G. Szymanski // Journal of the ACM. – 1982. – Vol. 29, Issue 4 . – P. 928–951. doi: 10.1145/322344.322346
9. Elias, P. Universal codeword sets and representations of the integers [Text] / P. Elias // IEEE Transactions on Information Theory. – 1975. – Vol. 21, Issue 2. – P. 194–203. doi: 10.1109/tit.1975.1055349
10. Rice, R. F. Adaptive Variable-Length Coding for Efficient Compression of Spacecraft Television Data [Text] / R. F. Rice, J. R. Plaunt // IEEE Transactions on Communications. – 1971. – Vol. 16, Issue 9. – P. 889–897. doi: 10.1109/tcom.1971.1090789
11. Левенштейн, В. И. Избыточность и задержка восстановительного кодирования натуральных чисел [Text] / В. И. Левенштейн // Проблемы кибернетики. – 1968. – № 20. – С. 173–179.
12. Golomb, S. W. Run-length encodings [Text] / S. W. Golomb // IEEE Transactions on Information Theory. – 1966. – Vol. 12, Issue 3. – P. 399–401. doi: 10.1109/tit.1966.1053907
13. Even, S. Economical encoding of commas between strings [Text] / S. Even, M. Rodeh // Communications of the ACM. – 1978. – Vol. 21, Issue 4. – P. 315–317. doi: 10.1145/359460.359480
14. Klein, S. T. On the usefulness of fibonacci compression codes [Text] / S. T. Klein, Kopel Ben-Nissan // The Computer Journal. – 2010. – Vol. 53, Issue 6. – P. 701–716. doi: 10.1093/comjnl/bxp046
15. Bastys, R. Fibonacci Coding Within the Burrows-Wheeler Compression Scheme [Text] / R. Bastys // Electronics and Electrical Engineering // Kaunas: Technologija. – 2010. – Vol. 1, Issue 97. – P. 28–32.
16. Somasundaram, K. Compression of Image using Fibonacci Code (FC) in JPEG2000 [Text] / K. Somasundaram, P. Sumitra // International Journal of Engineering Science and Technology. – 2010. – Vol. 2, Issue 12. – P. 7311–7319.
17. Калошин, Д. Б. Сравнение кодов переменной длины: коды Элиаса и Фибоначчи применительно к вопросам сжатия даннях [Текст] / Д. Б. Калошин, В. Ю. Башкирев, Е. В. Бурмин // Сейсмические приборы. – 2008. – Т. 44, № 3. – С. 64–69.
18. Анисимов, А. В. Линейные формы Фибоначчи и параллельные алгоритмы большой размерности [Текст] / А. В. Анисимов // Кибернетика и системный анализ. – 1995. – № 3. – С. 106–115.
19. Лужецький, В. А. Спосіб зображення цілих чисел великого діапазону [Текст] / Лужецький В. А., Мохаммад Аль-Майта. // Вимірювальна та обчислювальна техніка в технологічних процесах. – 1998. – № 1. – С. 156–162.
20. Кшановський, О. Д. Арифметичні методи ущільнення цифрової інформації [Текст] / О. Д. Кшановський, С. В. Тітарчук, В. А. Лужецький // Вісник ВПІ. – 1999. – № 5. – С. 83–87.